WEB APPLICATION DEVELOPMENT TOOL

TECHNICAL FIELD

[1] The present invention relates generally to the development of web applications and, in particular, to merging independently developed modules into an integrated web application.

BACKGROUND ART

[2] Large software projects are frequently divided into separate components to be completed by independent development teams. For example, in Java Enterprise Edition (J2EE) projects, applications are typically divided into a presentation or user interface component, a business logic component and a data component, each completed by an independent development team. Separating the development of an application in this manner reduces the complexity of development by isolating each team from the effects of changes in other parts of the project implemented by other teams.

[3] However, such a division of development effort may still be subject to unnecessary complexity in very large projects. Upon completion of the J2EE components, the presentation and business components are generally combined into a single archive (WAR) file. Although the two components could be packaged into multiple WAR files, separate packaging is not preferable because each WAR file has its own configuration data and/or runtime resources. Thus, WAR files will not be able to share resources, such as a context root (configuration data) or a session (runtime resource), with other WAR files.

[4] Consequently, a need exists to independently develop the separate components of a project while integrating the components into a single logical component at runtime.

SUMMARY OF THE INVENTION

[5] The present invention provides method, system and computer program product to increase the efficiency of the development of Java Enterprise Edition (J2EE)

applications. A project may be divided into modules which may be developed by independent teams. The files within each module are classified as independent of resources in other modules or dependent. Independent files may be packaged into a single, integrated web application archive (WAR) file without further processing. Corresponding dependent files are compared and any conflicts are resolved. The resulting files may then be packaged into the WAR file.

[6] Changes made to a module during development may be process and tested against modules previously integrated into the WAR file.

[7] Similarly, subsequent revisions to a module may be incorporated into the WAR file after first being processed to resolve any conflicts with files in the existing modules.

BRIEF DESCRIPTION OF THE DRAWINGS

[8] Fig. 1 is a representation of files available for an exemplary J2EE development project;

[9] Fig. 2 is a representation of the files of Fig. 1 incorporated into independently developed modules;

[10] Fig. 3 is a block diagram of the merge tool of the present invention; and

[11] Fig. 4 is a flow chart of the merge process of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[12] Fig. 1 is a representation of some of the resources, such as files File A 102, File B 104, File C 106, File D 108 and File E 110, which are available for an exemplary J2EE development project; it will be appreciated that many additional resources are typically present in a project. The project is divided into modules to be developed independently by different teams. Two such modules 200 and 220 are illustrated in Fig. 2; it will be appreciated that many additional modules are typically present in a project. The development team of the first module 200 has incorporated copies of some of the files: A 102, C 106, D 108 and E 110; the copies have been designated files 1A 202, 1C 206, 1D 208 and 1E 210. The development team of the second

module 220 has also incorporated copies of files C 106 and D 108 as well as copies of file B 104; these copies have been designated files 2B 204, 2C 206 and 2D 208.

[13] Referring to the block diagram of Fig. 3 and the flow chart of Fig. 4, after the teams have completed their modules, all of the resources (files 1A, 1C, 1D, 1E, 2B, 2C and 2D) are received in a first section 302 of the merge tool 300 of the present invention (step 400). In the first section 302 the resources of each module are examined to determine which are independent and do not correspond to any resource in a different module (step 402). An example of an independent file might include an XML file which defines validation constraints for a STRUTS framework. In Figs. 2 and 3, files 1A 203 and 1E 210 are used only in the first module 200 while file 2B 202 is used only in the second module 220. Thus, these three files are independent. The independent files are processed by a second section 304 of the tool 300 (step 404) and packaged into the application's WAR file (step 406).

[14] The remaining resources used in the first module 200 are each related to a corresponding resource in the second module: file 1C 206 is related to file 2C 226; file 1D 208 is related to file 2D 228. For example, a file to map error codes to error messages might be used in two modules.

[15] In a third section 306 of the tool 300, the related files are compared (step 408). The information in two corresponding files (such as files 1C and 2C) is examined to determine if any conflicts exist between the two. If no conflicts are present, the two files are merged (step 410) and passed to the second section 304 to be packaged in the WAR file 308.

[16] If a conflict in the information of two resources is identified, it must be determined whether the conflict is major or minor (step 412). If minor, a user may be notified with an appropriate message 310 (step 414), allowing the user to decide how to proceed. While less preferable, the information in one file may alternatively be ignored (step 416), displaced by the information in the other file.

[17] If the conflict is irreconcilable and, therefore, major, the user may be notified with an error message 310 (step 418), allowing the user to decide how to proceed. Alternatively, the process may abort (step 420), giving the development groups an opportunity to resolve the conflict by revising one or both of the related modules.

[18] After conflicts have been resolved, the merged files (files C and D in the example) are passed to the second section 304 of the tool 300 to be packaged in the WAR file 308 (step 406). The WAR file 308 is then ready to be installed into a J2EE application server as a J2EE application.

[19] During development, a development team may want to make revisions to a file and determine if the revisions will perform as expected. The module in which the revisions were made may be processed by the tool 300 and tested against other modules already integrated into the WAR file 308.

[20] Similarly, after development, revisions or updates to a file may be made. The module in which the revisions were made may be processed by the tool 300 and integrated into the WAR file 308.

[21] The objects of the invention have been fully realized through the embodiments disclosed herein. Those skilled in the art will appreciate that the various aspects of the invention may be achieved through different embodiments without departing from the essential function of the invention. The particular embodiments are illustrative and not meant to limit the scope of the invention as set forth in the following claims.